"Express Mail" mailing label no: EL653538504US Date of Deposit: January 24, 2001

ONX OPERATING SYSTEM NETWORK AUTO CONFIGURATION

BACKGROUND OF THE INVENTION

1 This invention relates generally to the field of computers and, more particularly, to the networking of computers for performing related tasks.

2 In a computer network utilizing Ethernet communications, it is necessary to configure the computers in the network to establish communications with each other and perform functions unique to their locations within that network system. In the past, it was necessary to configure a new or a replacement computer when placing it in the system. This required effort by a skilled system administrator. Additionally, computers already in the system would need to be reconfigured in order to communicate with any new or replacement computers.

3 As a result of the development of large systems which utilize computer controlled mechanisms, the control system required a network of computer nodes, which could communicate efficiently between themselves. Utilizing the QNX real time operating system, with its native networking environment (FLEET™, referenced in http://www.gnx.com/products/os/gnxrtos.html# networking) appeared to be the best alternative. A system manager computer was also attached to the network to provide a single point of interaction to most of the system functions. This manager computer was primarily to provide the operator with a Graphical User Interface (GUI) which would display the system status, allow the operator to start and stop the machine and also generate reports of past machine activity. The problems associated with such a system, however, is ease of maintainability, software installation, network configuration and hardware maintainability.

4 It is therefore an object of this invention is to provide a system of associating a list of tasks with a specific software for performing the specific task

5 It is another object of this invention to provide a system in which each node determines what location ID is to be used to download the appropriate application software to perform the task required at that location.

SUMMARY OF THE INVENTION

6 The objects set forth above as well as further and other objects and advantages of the present invention are achieved by the embodiments of the invention described hereinbelow.

7 For flexibility in machine configuration, and ease of maintainability (both from a software and hardware viewpoint,) the present invention utilizes the same controller for each section of the machine. Also, all software upgrades and configuration changes are performed from the master, with no direct interaction with each individual controller. Each controller has only a bare minimum of software installed on a Disk-on-Chip™ which would allow it to boot. This means that any controller for this machine could be swapped for any other controller (reducing the number of spares that would need to be maintained) and the machine would still function as intended. Each computer of a number of networked computers has an attached hardware controller unit for producing a hardware node ID.

8 Utilizing a specific hardware node ID, each computer can determine which node it is intended to act as, and which application or task program is needed to be executed at its associated site. Each computer also has a logical ID associated with it, which is determined at boot time, as a component of the QNX operating system. Such logical node ID is compared with the

hardware node ID of the computer, and should there be a mismatch, software reconfiguration occurs. During software reconfiguration, the appropriate operating system boot program is inserted into the computer having the proper logical node ID equal to the hardware node ID. The computer is then rebooted to load this portion of the operating system into memory.

9 For each computer, the logical node ID is also compared with the logical node ID associated with the physical node of the computer as indicated by a network through a mapping table, and with a physical address of the computer as indicated by a network adapter, to ensure that the each particular application program is being run at the proper physical site and to ensure that proper FLEET™ communications can be established between nodes. Also, the number of nodes that should be on the network as indicated by a system manager configuration file is compared with the number of nodes actually on the network to ensure that the proper number of nodes are network operative.

10 When computers running with the QNX operating system boot up, they have resident software that will read a configuration switch that is external to the computer. After reading the switch, the hardware node ID is used as the host ID portion of the IP address. Once the small board computer (the PC) is configured with the correct IP address, communications (via TCP/IP, not FLEET™) with other computers on the network can begin. Each QNX node will access the system manager hard disk to access executables and files using the SMB (Server Message Block) file sharing protocol. If the hardware node ID does not match the current software logic node ID as specified by the QNX system, then the proper QNX files will be retrieved from the system manager and the QNX node will then reboot itself. At this point, each QNX node will execute the proper software to perform the proper functions for its location as indicated by utilizing the unique logical node ID as a reference pointer to a

R.F. 1/22/2001

W.D.L. 1/22/2001

configuration file on the system manager hard drive. The configuration file will indicate what software needs to be loaded by the node from the system manager hard disk. With the executables and configuration files on the system manager hard disk, future software and/or hardware upgrades can be implemented without changing resident software on a QNX node.

11 For a better understanding of the present invention, together with other and further objects thereof, reference is made to the accompanying drawings and detailed description and its scope will be pointed out in the appended claims.

BRIEF DESCRIPTION OF THE DRAWINGS

12 Figure 1 is a schematic flow chart of the target system of this invention which illustrates the network connectivity for the invention as well as node ID input;

13 Figure 2 is a schematic flow chart of the communication protocol between nodes in the target system of this invention; and

14 Figures 3 and 4 comprise a flow chart describing an algorithm used to implement the invention.

DETAILED DESCRIPTION OF THE INVENTION

15 Reference is now made to Figure 1 of the drawings which illustrates the network connectivity for the invention as well as the node ID inputs. The QNX operating system addresses each computer in its network as a node. Each node has a unique logical ID, starting at 1. The node ID is normally established at the time the operating system (OS) is installed on the computer, and stored in a ".boot" file in the root directory, which is the first file loaded for the operating system. If the

node ID is to be changed, this file must be rewritten and the computer rebooted.

16 In order to better understand the overall operation of the invention in general terms, an example of a robotics assembly line, will be utilized for illustration but not for limitation, in connection with Figure 1. The assembly line can include robots that obtain an assembly part, clean and orient it, weld it and test the integrity of the weld. As stated above, this example is only illustrative of the inventive concept and should not be used as a limitation for other uses of the invention.

17 In the illustrated application there will be a computer control system made up of the following: single system manager PC 1 coupled to all QNX nodes such as nodes 3 and 5 and at least several other nodes not shown. Computer node 3 has a PC, which will execute a first applications program to operate a robot, for example. At a first cleaning station site also denoted as 3, a first robot is provided for cleaning the assembly prior to welding it at station site 5. The applications control program in the first node PC 3 has the instructions to enable the first robot to effect the cleaning operation. This first application will be associated with a unique logical node ID (in a configuration on the system manager hard drive) assigned to station site 3.

18 Once the cleaning operation is complete, the assembly moves on,for example, a conveyor (not shown) to a second robot at station site 5 that welds seams on the assembly. This second robot is controlled by computer node 5 and therefore denoted by the same reference numeral. A second task application program in this node 5 computer controls the welding robot at station site 5. In a like manner , for example, a third robot (not shown) can be used to check the integrity of the weld and is controlled by a third computer node (not shown) As pointed out above, other

operations and systems can be substituted for the robot example still within the confines of this invention. System manager 1 receives progress updates from the computer control system and stores the task application control software for each control computer.

19 As each control computer initializes, it reads an associated hardware switch that is set to a unique value which is used to identify the type of robot operation that the computer will have to control. This is the hardware node ID, which could be Node ID-1 for the first station site described. This allows the site PC to determine which application program is needed to be executed. The logical node ID for the welding task software application control program at site number 5 could be assigned logical Node ID-2 and so forth for the software application programs for subsequent tasks to be performed at the remaining station sites. As each computer node reads the value of the logical node ID off the associated hardware switch, it then loads the appropriate control software from the hard disk of the system manager PC that is associated with the logical node ID number. The logical node ID number is conveniently established by manually setting the hardware switches. Such action allows each control PC to be of the same type and configuration (meaning it would be the same assembly part number, or the same off the shelf computer, all configured exactly the same). As future requirements demand additional robot stations, the same control PC could be used and the specific application software would be downloaded from the system manager. Replacement of the control PCs would not require specialized system administration. The replacement control PCs can be physically connected and powered up.

20 As described above, each node has a specific set of tasks it must complete, dependent upon its site location within the control system. Therefore, there must be some way of associating

a list of tasks with a site location, and a way for each node to determine which location it is responsible for monitoring and controlling. The mapping between logical and physical node ID's is performed by the network manager. The logical node ID must be associated with the physical ID of the nodes network adapter. In the above example, if the computer at the first node (cleaning site 3 in Figure 1), was previously installed and configured for use at welding site 5, this erroneous situation must be corrected. This leads to a mismatch between the logical ID and the physical node Id which will be detected. The logical ID is then changed to match the hardware ID. Cleaning site 3 and welding site 5 will reboot and load and execute the appropriate application programs.

21 Every network card manufactured has an unique 12 digit hexadecimal (i.e. 0000C0 9A7F4C) address associated with it. Each node must know of its logical to physical ID mapping, as well as the mapping for all other nodes. The logical-to-physical mapping is usually contained in a file which is normally created at the operating system or OS installation. If this mapping is not correct (on all nodes,) FLEET™ communications between nodes cannot be established. If a node's ID must be changed, this file must be rewritten on all nodes. All nodes must then be rebooted, or at least have a system administrator level command run on each node as illustrated in the flow chart Figures 3 and 4.

22 Preferred embodiments of the present invention are based on network capabilities provided in the QNX operating system version 4.24 or higher. It also requires a system manager (or master) computer and its operating system, which is capable of supporting TCP/IP (Transmission Control Protocol/Internet Protocol) communications, with SMB (Server Message Block) or NFS (Network File System) file-sharing protocol, over an Ethernet connection as indicated in Figure 2. The system manager would provide storage for system files and application software to be

downloaded by the QNX nodes. QNX nodes are any PC-AT compatible platform capable of accepting node ID input via any input method.

23 The system files stored on the system manager 1 include node specific files, such as ".boot," and node independent files such as ".licenses" (QNX license file) and "sysinit". The system manager is also used for storage of configuration information such as the "netmap files". Application software stored on the system manager is based on the functional requirements of each node. One necessary configuration file maintained on the system manager is a file with a single numeric value stored in it to indicate how many nodes should be on the QNX network.

24 Each QNX node must also have a method of identifying which node it is intended to act as. This could be accomplished by installing the aforesaid hardware switch which could comprise a DI/DO (Digital Input/Digital Output) card and reading DIP (Dual Inline Pin) switches. As mentioned above, the switch settings represent a logical node ID that is unique for each node within the network.

25 As each QNX node initializes, or boots, it executes an initialization utility, called "sinit", which reads commands from a file called "sysinit.x" (where x is the node ID). First are commands which need to be executed on all nodes, including some high-level network initialization. Then, by executing a command to query the hardware, it is determined what the intended node ID is. The intended node ID (nid) is then used to establish TCP/IP communications by using it as the host ID portion of a TCP/IP address. (xxx.xxx.xxx.nid) Once this is accomplished, the system manager is contacted and its file system is mounted as a virtual device by the QNX node. If SMB is the file-sharing protocol, then this would be accomplished by spawning SMBfsys on the QNX node then executing mount_smb, otherwise, the mount would be accomplished by executing the mount_nfs command.

26 After initial communication is established, the hardware node id is compared with the logical node id contained within the ".boot file". If the two do not match, then steps must be taken to reconfigure the system to match the intended (or hardware) node id. These steps include copying the correct ".boot" file from the system manager and ensuring the "sysinit.nid" file (or a symbolic link) exists. If the hardware Node ID #1 calling for cleaning of the assembly in the computer at the first site, and the application program in this computer is logical ID #2 calling for welding of the assembly, the welding program must be replaced by the cleaning program.

27 Regardless of whether or not the IDs match, the "sysinit" file is compared with a copy maintained on the system manager. If there are differences, then the copy from the system manager is copied to the correct location on the local node. This allows for upgrades/changes to be implemented by modifying the file on the system manager alone, and the changes then getting propagated to the QNX nodes. If either the ".boot file" or "sysinit" file were changed, the QNX node is then automatically rebooted for the changes to take effect, and the above steps are repeated.

28 Once the hardware node ID and the logical node ID match, processing continues by querying the network adapter for its physical address. This information is then written to a file on the system manager named "netmap.nid", along with the necessary information for QNX to associate this physical ID and the nodes logical ID. At this point, all QNX licenses are also read into memory, from the system manager.

29 After this file is written, the netmap utility is executed, with input from all the netmap.nid files that exist on the system manager, to establish FLEET™ communications between nodes by establishing the logical to physical mapping for each node. When

execution has completed, a check is done to establish how many nodes have been initialized. A comparison is then made with the value stored in the configuration file on the system manager to see if all necessary nodes are initialized. If they are not, then a loop is entered to re-execute the netmap command, after a slight pause, until the number of initialized nodes matches (or exceeds) the configuration file value as illustrated in the flow chart of Figures 3 and 4.

30 When all nodes are initialized, processing continues as each node downloads and executes node specific application software as indicated in files (similar to the "sysinit" file) on the system manager to indicate what tasks each needs to perform.

31 The following description relates to the preferred system for interconnecting the computer nodes and includes references to FLEET™ which as pointed out above was obtained from the internet at http://www.qnx.com/products/os/qnxrtos.html#Networking .

32 Regarding the FLEET™ - High-performance Networking system for QNX, a unique feature of the QNX real-time operating system, FLEET™ creates a single homogeneous set of resources that you can access transparently, anywhere throughout the network. FLEET™ is an ultralight, high-speed networking protocol. Its innovative and feature-rich design turns network-connected machines into a single logical supercomputer. Because FLEET™ is built on the message-passing architecture of the QNX OS, it offers the ultimate in flexibility. FLEET™ delivers:

Fault-tolerant networking
Load-balancing on the fly
Efficient performance
Extensible architecture
Transparent distributed processing

33 Regarding fault-tolerant networking: If a cable or network

card in one network fails, FLEET™ automatically re-routes data through another network. This happens on the fly, without involving application software, giving you automatic network fault-tolerance.

34 Regarding load-balancing on the fly: Network throughput is normally limited by the speed of the computer used and network hardware. With FLEET™, data can be transmitted over multiple networks simultaneously, allowing a user to double, triple, or even quadruple network bandwidth and throughput by placing multiple network cards in each computer and connecting them with separate cables. It is even possible mix different types of network cards (e.g. Ethernet, FDDI) in the same machine.

35 Regarding efficient performance: FLEET™ network drivers are built to make the most of networking hardware. For example, when sending large blocks of data over an Ethernet network from one process to another, impressive throughput is obtained.

36 Regarding extensible architecture: As a result of FLEET™, a QNX network provides unsurpassed flexibility. Networking processes are architecturally distinct from the OS, allowing a user to start and stop a networked node at any time. This means a user can add nodes to a network or remove them dynamically without reconfiguring the system. And, as a result of automatic network bridging, you can add different physical networks to your LAN as well.

37 Regarding transparent distributed processing: FLEET™'s networking processes are deeply integrated into the heart of message-passing and process-management primitives, making local and network-wide IPC one and the same. Since IPC is network transparent, a network of individual PCs appears as a single, seamless supercomputer. The end result is that the user never needs to modify the applications to communicate across the

network.

38 Regarding file and term explanations: SMB(Server Message Block) is a file sharing protocol, which is used by a number of different servers (including: Windows NT, Windows 95, Windows for Workgroups, LAN Manager, and Samba). This protocol allows a QNX client to transparently access remote drives residing on such servers.

39 "The physical node ID is determined by the hardware. Network cards communicate with each other by specifying the physical node ID of the remote node they which to talk to. In the case of Ethernet and Token Ring, this represents a large number that is difficult for people and utilities to deal with. For example, each Ethernet and Token Ring card is shipped with a unique 48-bit physical node ID, conforming to the IEEE 802 standard." (This excerpt is taken from the QNX Operating System - System Architecture manual.)

40 The logical node ID: "To overcome the problems with physical node ID's, each QNX node is given a logical node ID. All QNX processes deal with logical node ID's. The physical node ID's are hidden from processes running on QNX. Logical node ID's simplify network and applications licensing. They also make it easy for some utilities that may wish to poll the network using a simple loop, where the logical nod ID goes from 1 to the number of nodes." (This excerpt is taken from the QNX Operating System - System Architecture manual.)

41 The mapping between logical and physical node ID's is accomplished by the network manager. The driver is given a physical ID by the network manager when asked to transmit data to another node. The logical node ID is typically assigned sequential numbers starting at 1. For example, a node with an Ethernet card may be given a logical node ID of 2, which is

mapped to the physical node ID of 00:00:c0:46:93:30. Note that the logical node ID must be unique for all nodes across all interconnected QNX networks in order for network bridging to work."

42 Regarding the logical network ID: "The network ID identifies a particular logical network. A logical network is any hardware that allows a network driver to directly communicate with a network driver on another node."

43 (Excerpt taken from QNX Operating System - System Architecture manual) In the present invention, there is only one network in the control system. Therefore the logical network ID was set as 1 for all nodes.

44 Regarding the ".boot file": This is an executable file that runs when the computer is started. It is an image of the operating system. There is a dependency on the logical node ID in this file. That is why there are specific versions of this file on the system manager hard disk. Therefore as a QNX node compares the hardware node ID with the Logical node ID and they are different, a new .boot file must be obtained from the system manager and the QNX node restarted to execute the correct ".boot file".

45 Regarding the netmap file:"The netmap file is the default node and network ID mapping file used by the netboot and netmap utilities. This file defines the physical node ID's, the logical node ID's, and the logical network ID's (LAN) associated with each node." (Excerpt taken from QNX Operating System - Installation and Configuration manual)

46 An example file is shipped with QNX:

#Logical LAN Physical

1	1	0000c0	9a7f4c
2	1	000629	d07830
3	1	0000c0	454e79

47 Regarding the sysinit file:"When QNX boots, an image (the ".boot" executable) composed of several QNX processes is loaded into main memory. The first process is boot. (Excerpt taken from QNX Operating System - Installation and Configuration manual)

48 The second process in the image is the Process Manager (Proc32), which contains the Microkernel. The last process in the image is the sinit utility. The sinit utility initiates the second phase of system initialization by starting a shell that executes commands a file. This file, the system initialization file (sysinit) contains commands that set up services for a machine.

49 Being able to start system services after boot is one of the benefits of QNX's modular architecture. The booted image typically contains only the few essential services needed to start all other required services. During a normal boot, sinit tries to boot from the "sysinit.node" file. Copies of the "sysinit" files are kept on the system manager. There are lines in the "sysinit" file that check to see if the copy on the system manager is different than the local copy on the QNX node. Therefore any updates to this file, as well as others, can be placed on the system manager station and are automatically utilized by the QNX nodes after the nodes are booted as illustrated in the flow chart of Figures 3 and 4.

50 Although the invention has been described with respect to various embodiments, it should be realized this invention is also capable of a wide variety of further and other embodiments within the spirit and scope of the appended claims.

51 What is claimed is: